Concesionario de Naves Espaciales

Creadores: -Alejandro López Adrados

-Guillermo Grande Santi

-Adrián Morales Dato

-Nicolás Rubira Cabello

En esta memoria vamos a explicar cómo hemos realizado la práctica del concesionario de naves espaciales de segunda mano. Iremos explicando clase a clase su funcionamiento y cómo se han desarrollado los métodos.

La clase **Main** simplemente inicia el programa y crea un objeto de tipo **ClientManager**. El main es un while(true) que volverá a iniciarse por cada vez que termine una sesión.

En esta última clase, se crearán los ficheros de naves y clientes en el caso de que no estén creados, y se creará un objeto de la clase **Login**, llamando a su único método en el que el usuario podrá iniciar sesión con su correo y su contraseña. Si la contraseña es fallida tres veces se volverá a la página de inicio de sesión. Una vez se haya iniciado sesión correctamente, se evaluará el tipo de usuario con el método de la clase Admin (explicado posteriormente). Si el usuario es de tipo 0 (más de dos advertencias o fraudulento) no podrá entrar al sistema (esto se explica posteriormente en los métodos de la clase Admin). Después de esto, se crea un objeto de tipo SuscriptionManager y se comprueban las ofertas nuevas a las que estaba suscrito ese cliente (mirando su último acceso). Por último, se creará un objeto de tipo **OperationMenu** y se llamará al método printMenu() para elegir la operación, pasando como parámetros el Tipo de Usuario y el Cliente que ha iniciado sesión.

En este menú encontraremos 8 opciones entre las que podremos elegir.

- La **opción 1** creará un objeto de tipo BuyOffer y llamará a su método público verOfertas(). Esta clase es de las más densas y

largas, ya que necesitamos un método para mostrar las diferentes ofertas publicadas, otro para seleccionar los filtros que queremos en la búsqueda de ofertas, otro para seleccionar la oferta que se desee comprar, y un último para comprar la oferta. Los tres últimos serán privados. Resumiendo, en primera lugar se crearán las variables necesarias y se seleccionarán los filtros. Para ello se podrá elegir cuántos y cuáles filtros poner a la búsqueda y devolveremos una lista de strings. Una vez devuelta, se escribirán por pantalla todas las naves de cada oferta en la lista de ofertas publicadas, pero para ello habrá que haber comprobado el Tipo de Usuario antes. Si es de tipo 1, solo podrá ver ofertas con Cargueros. Si es de tipo 2, sólo podrá ver ofertas con Cargueros y Cazas. Si es de tipo 3, podrá ver todas las ofertas. Una vez conseguida las ofertas válidas para cada usuario éstas se guardarán en una lista, y se llamará a seleccionarOferta(). Gracias a este método, podremos elegir una entre todas las ofertas de la lista de ofertas válidas. Este método nos devolverá la oferta elegida y llamaremos a comprarOferta(), que servirá para crear una transacción en la que se guardará el email del comprador, del vendedor, la fecha en la que se realizó la transacción y la oferta que se compra. Además, la nave pasará a pertenecer al comprador. Una vez devuelta la transacción, se usará el método addTransaction de Admin para añadirla a el registro de transacciones (fichero), y por último se pedirá al usuario una valoración (comentario y votación).

- La **opción 2** creará un objeto de tipo CreateOffer y llamará a su método crearOferta(). Este método mostrará primero tus naves disponibles (gracias al método getShipList() de Admin). Con un método privado llamado seleccionarNaves(), se seleccionarán las naves que deseas vender entre todas las que tienes. Este método devuelve una lista de String (los números de registro de las naves que quieres añadir a la oferta). Finalmente, indicarás el precio, descripción, y fecha límite de la oferta, se creará y se añadirá al fichero Ofertas con el método addOffer() de Admin.

- La **opción 3** creará un objeto de tipo SubscribeToShip en el que su método susbcribe() permitirá al usuario elegir el tipo de nave al que

se quiere suscribir, llamando a admin.setSuscriptor() para cambiar su estado de suscripción en el fichero.

- La **opción 4** creará un objeto de tipo UnsubscribeToShip en el que su método unsusbcribe() permitirá al usuario elegir el tipo de nave al que se quiere desuscribir, llamando a admin.setSuscriptor() para cambiar su estado de suscripción en el fichero.

- La **opción 5** creará un objeto de tipo LastTransactions en el que su método lastTransaction() mostrará las últimas transacciones (máximo 10). Se mostrarán tanto las ventas como las compras. Para ello se busca el cliente con el mismo email en el fichero y se usa el método getIndividualTransaction() de Admin para conseguir las transacciones de ese cliente.

- La **opción 6** creará un objeto de tipo MyRatings y llamará a su método seeRatings(), en el que se mostrarán los comentarios que se han dejado a ese cliente en las anteriores compras y una media de todas las valoraciones que tiene. Ambas cosas se guardan en un solo String, pero siempre el primer subString (0,1) está formado por un entero que es la valoración numérica, por lo que se puede sacar y utilizarlo por separado.

- La **opción 7** creará un objeto de tipo MyProfile y llamará al método myAccount(), que simplemente mostrará información de tu perfil como tu nombre, tu email, tu id, tu fecha de último acceso, el número de advertencias que tienes y una lista de tus suscripciones activas en el momento.

- Por último, la **opción 8** servirá para salir del bucle en el que se elige opción y se volverá al main (volviendo a empezar el programa).

Además el programa contiene diferentes clases que no son operaciones del usuario pero que sirven para su ejecución:

La clase **FileManager** guarda la operativa con los ficheros de todo el programa. En ella se encuentran los métodos necesarios para escribir, modificar y eliminar los ficheros de ofertas, transacciones, clientes y naves:

- Escribir en fichero: Para escribir en un fichero es necesario profundizar en la clase ObjectOutputStreamMod. Esta clase lo que permite es escribir diferentes objetos sin "machacar" el anterior. Una vez definida esta clase, se comprueba si el fihcero existe y si es así llama al método escribir objeto de la clase anteriormente explicada.
- Leer en fichero: Para leer los diferentes objetos de un fichero, se crea un objeto de ObjectInputStream del cual se utiliza el metodo readObject() esto se realiza en un bucle infinito hasta que se captura la excepción EOF(fin del fichero). Para la correcta lectura del objeto en cuestión hay que hacer un casting a diicho objeto.
- Modificar objeto de un fichero: Para modificar cualquier atributo de un objeto determinado hay que buscar ese objeto en su correspondiente lista. Una vez encontrado la referencia a esa lista se modifica y se vuelve a escribir el fichero entero "machacando" lo anterior.

Estos ficheros se almacenan en la raíz del programa con la extensión .dat. Sin embargo, esta clase no es utilizada por el resto de clases del programa sino que solo es usada por la clase Admin.

La clase **Admin** contiene los métodos que se usan a lo largo de todo el programa relativo a los ficheros, en esta clase se encuentran métodos que devuelven las listas de los objetos leídos en los diferentes ficheros y métodos para modificar diferentes valores de cualquier objeto. Todo esto lo realiza usando los métodos del FileManager. Además, incluye el método evaluarTipoUsuario el cual devuelve un entero basándose en las restricciones de cada cliente (piratería, fraude, advertencias...).

Otra de las clases que no se muestran en el menú es la clase **FilesCreator**. Esta clase es la encargada de crear los ficheros y guardar en ellos los objetos para el primer uso, ya que como se explica en el manual, no se permite crear usuarios ni naves sino que se trabaja con los usuarios ya registrados y con sus correspondientes naves para vender.

La clase **SuscriptionManager** es la encargada de notificar al usuario en cada inicio de sesión si se han publicado nuevas ofertas que incluyen las naves a las que está suscrito. Para ello se registra cada inicio de sesión del usuario y se registra la fecha de publicación de cada oferta. Es por ello que esta clase revisa estos datos y si el usuario está suscrito a ese tipo de naves, la fecha de publicación de la oferta en cuestión es posterior a su último acceso y la oferta no es suya se añade al contador de ofertas por revisar. Este contador es el que le aparecerá al usuario en cada inicio de sesión.

Por último la clase **AdminMenu**, a la cual se accede mediante el usuario Admin y contraseña Admin. Esta Clase contiene un menú con diferentes métodos:

- Revisar ofertas: Se muestran las ofertas sin publicar al usuario (Obtenidas desde el fichero ofertas y eliminando las publicadas) y se deja seleccionar una. Tras esto la oferta seleccionada se despliega aportando más datos y se da la opción de publicar o eliminar. Si se publica se modifica el boolean de publicada y si se elimina se llama al método eliminar oferta y además se suma una advertencia.
- Añadir/borrar piratería/fraude: Muestra la lista de todos los clientes, se selecciona uno y se llama a su método correspondiente en admin y lo que hace este método es cambiar el booleano del cliente en cuestión y modificarlo en el archivo de clientes.

Por último este programa utiliza objetos que cuentan con propiedades pero no métodos.

La clase **Nave** es la que almacena todas las propiedades que puede tomar cada tipo de nave. De esta derivan los diferentes tipos (Carguero, caza, destructor y estación espacial). Sin embargo, la creación de estos objetos no se realiza mediante un constructor habitual sino que se crear a partir

de la interfaz creator. Esta interfaz es implementada por los diferentes subtipos de naves CargueroCreator, CazaCreator... y se utiliza para crear cada subtipo con las propiedades específicas ya que como indica el manual cada subtipo tiene unas propiedades en común y otras propiedades que las caracterizan. Esto se explicó más en detalle en el diseño UML con la explicación del patrón Abstract Factory.

Por otra parte también se encuentran otros objetos como clientes, ofertas y transacciones con sus respectivas propiedades (se intuye su estructura por lo explicado en las operaciones).

LISTA DE CLIENTES CON LAS QUE PUEDES INICIAR SESIÓN:

Puedes consultar la clase FilesCreator, pero esto es un resumen:

-Cliente 1 – Email: alelopez@gmail.com – Password: Alejandro – Tipo 3

-Cliente 2 – Email: Guillermo@gmail.com – Password: Hola – Tipo 1

-Cliente 3 – Email: NRC@gmail.com – Password: 0123 – Tipo 0

-Cliente 4 – Email: AMD@gmail.com – Password: 3366 – Tipo 3

-Cliente 5 – Email: kromi@gmail.com – Password: kromi – Tipo 2

Recomendaciones:

Esto no debería de pasar, pero en caso de que ocurra estos son algunos consejos.

- Si el programa vuelve al inicio sin que el usuario lo desee se ha crasheado.

- Si el programa se crashea en una operación determinada, puede ser porque el fichero esté corrupto. Para ello elimina todos y vuelve a ejecutar el programa de nuevo.

ENLACE PÚBLICO DEL REPOSITORIO:

https://github.com/Adri-md-1208/Concesionario-Naves